MUSST: A Multilingual Syntactic Simplification Tool

Carolina Scarton and **Lucia Specia**
University of Sheffield
Sheffield, UK

Alessio Palmero Aprosio and **Sara Tonelli**
Fondazione Bruno Kessler
Trento, Italy

Tamara Martín Wanton
H.I. IBERIA
Madrid, Spain

Abstract

We describe MUSST, a multilingual syntactic simplification tool. The tool supports sentence simplifications for English, Italian and Spanish, and can be easily extended to other languages. Our implementation includes a set of general-purpose simplification rules, as well as a sentence selection module (to select sentences to be simplified) and a confidence model (to select only promising simplifications). The tool was implemented in the context of the European project SIMPATICO on text simplification for Public Administration (PA) texts. Our evaluation on sentences in the PA domain shows that we obtain correct simplifications for 76% of the simplified cases in English, 71% of the cases in Spanish. For Italian, the results are lower (38%) but the tool is still under development.

1 Introduction

Text simplification is the task of reducing the lexical and/or syntactic complexity of a text (Siddharthan, 2004). It is common to divide this task in two subtasks: lexical simplification (LS) and syntactic simplification (SS). Whilst LS deals with the identification and replacement of difficult words or phrases, SS focuses on making complex syntactic constructions simpler. It is known, for instance, that passive voice constructions are more complex than active voice, and that long sentences with multiple clauses are more difficult to be understood than short sentences with a single clause. Several tools have been developed for LS (Paetzold and Specia, 2016). However, we are not aware of freely available tools for SS.

The SIMPATICO project[1] addresses text simplification for specific target audiences and domains. The project has three use cases focusing on different audiences: non-native speakers (Sheffield, UK), elderly (Galicia, Spain) and business and general citizens (Trento, Italy). Although personalised simplifications for each user type is our ultimate goal, we lack user-specific data. Therefore, our first step was to design general-purpose simplification rules which will later be specialised for the domain under consideration (PA). This solution led to the development of MUSST, which includes SS modules for three languages.

MUSST is based on the framework proposed by Siddharthan (2004) and is available as an open source Python implementation. Our rules split conjoint clauses, relative clauses and appositive phrases, and change sentences from passive into active voice. These are arguably the most widely applicable simplification operations across languages. We use the Stanford dependency parser (Chen and Manning, 2014) for the three languages, which enabled us to build a consistent multilingual tool. MUSST is evaluated using corpora extracted from the SIMPATICO use cases data. Such corpora (one for each language) were checked and – where applicable – syntactically simplified by experts in the area.

Inspired by the work of Gasperin et al. (2009), we also developed a complexity checker module in order to select sentences that should be simplified. In addition, we implemented a confidence model in order to predict whether or not a simplification produced by MUSST is good enough to be shown to the end-user. Developing these two modules required small labelled training sets.

To the best of our knowledge, MUSST is the

[1] https://www.simpatico-project.eu/

first freely available, open-source tool for SS in three languages, which – because of its modular nature – can be extended to other languages using the same framework.

2 Architecture

The architecture of MUSST has three main modules: analysis, transformation and generation (Siddharthan, 2004).

2.1 *Analysis*

The *Analysis* module is responsible for processing sentences to search for clues for the simplification of conjoint clauses (discourse markers) and relative clauses (relative pronouns). Sentences where cues are found trigger dedicated functions of the *Transformation* module.

Discourse markers need to be classified according to their semantics, in order to be correctly handled by the simplification rules. For instance, a conjoint clause with "and" as discourse marker should be processed differently from a conjoint clause with "when" as a discourse marker.

At this stage, the only mandatory text preprocessing steps are tokenization, which is done using the Stanford dependency parser, and lowercasing, which is done using the built-in function `lower()` in Python.

2.2 *Transformation*

In the *Transformation* module, our rules that simplify conjoint clauses, relative clauses, appositive phrases and passive voice are applied. This module is the core of MUSST. The main method is called `simplify`, which receives a sentence as input and returns one or more simplified sentences. The simplification is implemented as a recursive process that will keep simplifying the sentence until there is no more simplification rule that applies. The order of simplification is: appositive phrases, conjoint clauses, relative clauses and passive voice. This order has been defined empirically.

All the simplifications are done based on the output of the dependency parsers. For English and Spanish, we used the parsers available in CoreNLP[2], trained with Universal Dependencies[3] datasets. For Italian, we used the parser available in Tint[4] (Palmero Aprosio and Moretti, 2016) (an adapted version of CoreNLP for Italian).

Figure 1 shows the parser output for the sentence "These organisations have been checked by us and should provide you with a quality service.", as an example. The sentence is first sent to the *Analysis* module that will search for discourse markers. In this case, "and" is found and the sentence is thus sent to the conjoint clauses rule. Such a rule searches for two tags in the root dependencies: ADVL (adverbial clause modifier) or CC (coordinating conjugation). In our example, there is a CC relation between "checked" (the root) and "and". Since "and" is on the list of markers in the *Analysis* module, the next step is to search for a CONJ (conjunction) tag. In the example, there is a CONJ relation between "checked" and "provide". The conjoint clause rule is then applied and the sentence is split into two. Each sentence is then sent to the *Generation* module. The simplified sentence at this stage is "These organisations have been checked by us. And these organisations should provide you with a quality service." Then, each of these simplified sentences are sent again to the simplifier in a recursive manner.

```
det(organisations-2, These-1)
nsubjpass(checked-5, organisations-2)
aux(checked-5, have-3)
auxpass(checked-5, been-4)
root(ROOT-0, checked-5)
case(us-7, by-6)
nmod(checked-5, us-7)
cc(checked-5, and-8)
aux(provide-10, should-9)
conj(checked-5, provide-10)
dobj(provide-10, you-11)
case(service-15, with-12)
det(service-15, a-13)
compound(service-15, quality-14)
nmod(provide-10, service-15)
```

Figure 1: Example of parser output.

The *Transformation* module is also responsible for sending the *Generation* module all the information needed for re-generating the simplified sentences. This information includes: discourse marker, relative pronoun, PoS tag of main and modal verbs and PoS tag of subject.

2.3 *Generation*

This module is responsible for re-constructing the simplified sentence(s) and guaranteeing that gram-

[2] http://stanfordnlp.github.io/CoreNLP/
[3] http://universaldependencies.org
[4] http://tint.fbk.eu/parsing.html

maticality is preserved. It needs to account for the fact that a sentence can be split (conjoint and relative clauses and appositive phrases) or reordered (passive voice).

Truecasing and removal of extra punctuation are also implemented in this module. For truecasing, we call a Python implementation that has a pre-trained model for English[5], and train truecasing models using monolingual corpora for Spanish and Italian. For punctuation removal we use rules that identify punctuation repetition.

In the case of conjoint clauses, we may need to add specific discourse markers to the simplified sentences depending on the markers in the original one. For example, if the complex discourse marker is "although", the second simplified sentence will start with "but".

For appositive phrases, the verb that connects the subject to the apposition is defined according to the number of the subject and the tense of the main verb. For instance, the simplified version of "Truffles, a luxury food, are delicious." is "Truffles are delicious. Truffles are a luxury food.".

Changes in passive voice also require verb changes. Such changes need to respect the tense of the verb and the person number of the subject. Changes in the pronoun realisation are also modelled: when pronouns are the subject of the passive voice, they will become the object of the verb in active voice.For verb conjugation we use the NodeBox toolkit[6] for English and the tool for verb conjugation in Tint for Italian. For Spanish, we developed a new module.

No further treatment is needed for relative clauses.

3 Evaluation

For English, we selected $1,100$ sentences from the Sheffield City Council website[7]. Such sentences were processed by MUSST, which led to 292 simplified sentences. We categorised these simplified sentences depending on whether or not they were correct simplifications (according to grammar). From the 292 sentences, 70 sentences were considered incorrect. Errors are usually created from parser issues. For instance, the sentence "PE kit, school bag, packed lunch." was incorrectly simplified to "PE kit packed lunch. PE kit was school bag.". The dependency parser identified "packed" as the main verb, so the appositive phrase rule was applied. Since such problems are difficult to detect during simplification, we suggest using a confidence model (Section 4.2) after simplification.

For Italian, on a test set of 263 Italian sentences from SIMPITIKI corpus (Tonelli et al., 2016), 92 were simplified by MUSST. 57 of these sentences were judged as incorrect simplifications. The major cause of problems is also parsing errors, especially when sentences are particularly long or have ambiguous connectives.

For Spanish, out of 73 sentences from the Xunta Galicia website[8], 49 sentences were simplified by MUSST. Only 14 of these sentences were considered incorrect and the main issues were also due to parsing errors.

4 Extra modules

4.1 Complexity checker

Gasperin et al. (2009) proposes a binary classifier to decide whether or not a sentence should be split. We build a similar but more general classifier to decide whether a sentence should be simplified (including passive to active voice simplification).

We use the Naive Bayes implementation from the scikit-learn toolkit[9] to train a classifier with 10-fold cross-validation[10]. As features, we extracted simple counts of content words, syllables, tokens and punctuation along with number of clauses, discourse markers and relative pronouns.

For English, we used the $1,100$ sentences presented in Section 3. For Italian, we used a set of 405 sentences from SIMPITIKI and, for Spanish, we used a set of 104 sentences from the Xunta de Galicia website. All these sentences had been manually checked and – where applicable – simplified by experts, so each simplified sentence was considered a positive example.

Table 1 shows the performance of our classifiers in terms of precision, recall, F1 score and accuracy. All languages outperform the *majority class* classifiers in terms of accuracy (values in brackets), even though we rely on simple features and small training sets. The best F1 was achieved by the model for Spanish, closely followed by the model for English. Although the model for Italian

[5] https://github.com/nreimers/truecaser

[6] https://www.nodebox.net/code/index.php/Linguistics

[7] https://www.sheffield.gov.uk/

[8] http://www.xunta.gal/portada/

[9] http://scikit-learn.org/

[10] Other algorithms performed worse.

```
cscarton:simpatico_ss$ python __main__.py -l en -d ../tests/examples.en.toy -comp -conf
These organisations have been checked by us and should provide you with a quality service.
Suppose you later have a problem and need to use your contingency money . Then you can contact our customer accounts team to explain why you need this money .
The U.S. price is currently 18 cents a pound .  The U.S. price runs well above the world rate . So the 23,403 tons are still a lucrative target for growers .
The 23,403 tons are three quarters of the share .
If people have got in place proper effective safety measures, then naturally we are pleased about that.
Although both India and Pakistan announced troop withdrawals along the border, they both left their forces in Kashmir intact.
```

Figure 2: Example of MUSST usage.

has the best precision, its recall is the worst. The model for English has the lowest precision, but the highest recall. The Spanish model has similar values for precision and recall.

	F1	Precision	Recall	Accuracy
English	0.61	0.56	0.68	0.81 (0.78)
Italian	0.60	0.63	0.57	0.66 (0.58)
Spanish	0.62	0.61	0.62	0.76 (0.70)

Table 1: Performance of the complexity checkers.

4.2 Confidence model

In order to decide whether the simplified version of a sentence is "good enough" for a user, we trained a confidence model to classify a simplification as acceptable or not. Using the 292 sentences simplified by the English system and evaluated in Section 3, we built a confidence model for this language. The 70 sentences classified as incorrect (Section 3) were used as negative examples, whilst the remaining sentences received the positive label.

As features, we used the same basic counts as for the complexity checker (Section 4.1) along with language model (LM) probabilities and perplexity and grammar checking on the simplifications. KenLM[11] (Heafield, 2011) was used to extract LM features. A Python grammar checker was used for evaluating grammaticality[12]. The model was trained using the Random Forest implementation from scikit-learn with 10-fold cross-validation and achieved 0.80 of accuracy (F1/Precision/recall = 0.60/0.69/0.53), outperforming the MC classifier (accuracy = 0.61).

For Italian and Spanish, we also experimented with the datasets presented in Section 3, but the performance is worse because of the significantly smaller training sets. Nevertheless, both models outperform the majority class baseline in terms of accuracy. For Italian, our model achieved 0.80 of accuracy, against 0.66 for the majority class baseline. For Spanish, our model achieved 0.73 of accuracy (baseline = 0.55).

[11] https://github.com/kpu/kenlm

[12] https://pypi.python.org/pypi/grammar-check/

5 Demo outline

MUSST is available for download at https://github.com/SIMPATICOProject/SimpaticoTAEServer/tree/ijcnlp2017-demo. During the demo session we will present simplifications using both the command line (e.g. as in Figure 2) and the graphical interface of the SIMPATICO Dashboard (http://simpatico.fbk.eu/demo2/webdemo/index.html). We will demonstrate the use of `-comp` and `-conf` parameters that activate the complexity checker and confidence model, respectively.

Finally, we will discuss how a new language can be included into the tool.

Acknowledgments

This work has been supported by the European Commission project SIMPATICO (H2020-EURO-6-2015, grant number 692819).

References

D. Chen and C. D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP 2014*.

C. Gasperin, L. Specia, T. F. Pereira, and S. M. Aluísio. 2009. Learning when to simplify sentences for natural text simplification. In *ENIA 2009*.

K. Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *WMT 2011*.

G. H. Paetzold and L. Specia. 2016. Benchmarking Lexical Simplification Systems. In *LREC 2016*.

A. Palmero Aprosio and G. Moretti. 2016. Italy goes to Stanford: a collection of CoreNLP modules for Italian. *ArXiv e-prints*.

A. Siddharthan. 2004. *Syntactic simplification and text cohesion*. Ph.D. thesis, University of Cambridge.

S. Tonelli, A. Palmero Aprosio, and F. Saltori. 2016. SIMPITIKI: a Simplification corpus for Italian extracted from Wikipedia. In *CLiC-it 2016*.